The Deviation Attack: A Novel Denial-of-Service Attack Against IKEv2

Tristan Ninet, Axel Legay, Romaric Maillard, Louis-Marie Traonouez, Olivier Zendra

The Deviation Attack: A Novel Denial-of-Service Attack Against IKEv2

Tristan Ninet*†, Axel Legay‡, Romaric Maillard†, Louis-Marie Traonouez* and Olivier Zendra*
* Inria {tristan.ninet,louis-marie.traonouez,olivier.zendra}@inria.fr
† Thales SIX GTS France romaric.maillard@thalesgroup.com
‡ U.C. Louvain axel.legay@uclouvain.be

***Abstract*—In previous analyses IKEv2 has been shown to suffer from an authentication vulnerability that was considered not exploitable. By designing and implementing a novel slow Denial-of-Service attack, which we name the Deviation Attack, we show that the vulnerability is actually exploitable. We explain the attack's requirements, propose possible counter-measures and propose two possible modifications of the protocol, which both overcome the vulnerability.**

I. Introduction

Virtual Private Networks (VPN) have been used for decades by companies, governments and people. VPNs allow connecting two or more distant IP entities as if they were in a single Local Area Network (LAN). By "entity" we either mean a network or a single machine. Since the network between the entities may be non-trusted, connecting the entities often implies performing some encryption and some address translation on the IP traffic between them. In a company, an entity can among other things denote a traveling employee or a working site. In the military domain, an entity can among other things denote a soldier on the battle field or a military base. More recently, we have observed the rise of commercial VPN services for the greater public. Companies offer individual consumers to enter one of the companies' VPNs so that the consumers can browse the Internet as if they were in one of the companies' LANs. Many people use VPN services to spoof their location in order to access services denied to them based on their geographical location. Many people also use them to evade censorship, since VPN services allow them to hide the content of their packets between them and the VPN service company. VPNs thus have become a core technology in the modern secure Internet, and it is crucial that VPNs remain secure.

A VPN can be achieved using multiple technologies, among which the Internet Protocol security (IPsec) architecture [1] is widely used. Moreover, Internet Key Exchange version 2 (IKEv2) [2] is one of the main protocols used to set up IPsec VPNs. IKEv2 aims at guaranteeing mutual authentication of two peers, and at automatically generating the shared secret that will be of the communication's security warrant. Thus a secure VPN relies upon the security of IKEv2, and it is of the utmost importance that IKEv2 be secure.

Previous analyses of IKEv2 [3], [4] have shown that the protocol satisfies only a weak form of authentication. These analyses exhibit an execution trace of IKEv2 that violates strong authentication: the penultimate authentication flaw. However, this vulnerability was not considered a serious concern because it did not question the secrecy of the shared key generated by IKEv2.

In this paper, we start by giving some background on IKEv2 and previous analyses of the protocol in Section II. In Section III, we design a novel Denial-of-Service (DoS) attack against IKEv2 that exploits the penultimate authentication flaw. We call the novel DoS attack the Deviation Attack. The Deviation Attack bypasses all measures that were introduced in IKEv2 to resist DoS attacks. We thoroughly discuss the Deviation Attack's flow and details, and calculate the precise quantities that trigger the attack. To demonstrate the Deviation Attack very concretely, we implement it in Section IV; thereby attacking an open-source implementation of IKEv2. We experimentally verify our expression of the triggering quantities through this experiment, and provide the source code so that the reader can easily reproduce the attack.

Finally, in Section V, we explore a number of ways to protect the implementations only using what the current protocol specification has to offer. However, we only find mitigations or incomplete workarounds. We therefore tackle the problem at a higher level: We propose two possible inexpensive modifications of the protocol, which both prevent the attack.

For ethical reasons we informed our country's national security agency about the existence of the Deviation Attack. The security agency gave us some technical feedback as well as its approval for publishing the attack.

II. Background

A. The IKEv2 protocol

Internet Key Exchange version 2 (IKEv2) is the authenticated key-exchange protocol used in the Internet Protocol security architecture (IPsec). Its specification is managed by the Internet Engineering Task Force (IETF), and the current RFC is RFC 7296 [2]. The goal of IKEv2 is to allow two peers to dynamically negotiate cryptographic algorithms and keys in order to set up an IPsec communication. As such, it aims to guarantee mutual authentication of the peers and secrecy of the negotiated keys. An open-source implementation of IKEv2 is strongSwan [5].

The IPsec architecture provides security at the networking layer. It is defined in RFC 4301 [1]. IPsec defines a framework

to establish Virtual Private Networks (VPN). Depending on the underlying security protocol that is used, IPsec provides either integrity/authentication protection (AH protocol) or confidentiality and integrity/authentication protection (ESP protocol) of IP packets that are exchanged between two peers. IPsec also natively brings some protection against replay attacks (using this protection is however at the discretion of the receiver of an IPsec protected packet). To protect packets, IPsec peers set up Security Associations (SA) between them. A Security Association is a set of security parameters and keys on which two peers agree.

Setting up an SA requires that the peers share some encryption keys and involves adding entries to their Security Association Database (SAD). The keys can be manually put into the peers' databases, but the maintenance of such a configuration becomes cumbersome when the number of peers grows. To ease management of large VPN setups, it is much more efficient to rely upon dynamic negotiation of cryptographic material as defined by the IKEv2 protocol.

We call an *exchange* the association of a request message and a response message. IKEv2 consists of three main exchanges. The IKE_SA_INIT exchange performs initial setup of an IKE SA that will be used to protect subsequent exchanges of the IKEv2 protocol. During this exchange, peers agree upon cryptographic algorithms that should be used to protect further IKEv2 exchanges and establish some common cryptographic material by running a Diffie-Hellman protocol. The IKE_AUTH exchange authenticates the peers, validates the IKE SA and sets up a traffic SA (or Child SA). The authentication can be done using pre-shared key (PSK) or digital signature. It is a counter-measure to the Man-in-the-Middle attack that can be performed against the Diffie-Hellman exchange of IKE_SA_INIT. Finally, the CREATE_CHILD_SA exchange has two different purposes. First, it can be used for rekeying an IKE SA, i.e. for replacing an old IKE SA with a new one. In this case, its payloads for performing a Diffie-Hellman exchange (the key-exchange payloads) are mandatory. Second, it can be used to create a new traffic SA, or to rekey an existing one. In this case, the key-exchange payloads are optional. If not provided, the new keys are simply derived from the IKE SA's keys. Using the key-exchange payloads provides Perfect Forward Secrecy for the new traffic SA's keys, i.e. their secrecy will not be impacted by the compromise of the IKE SA's keys. The CREATE_CHILD_SA exchange is performed multiple times during the lifetime of an IKE SA.

B. Related work

In 1999, Meadows finds two authentication weaknesses in IKEv1 [6], using the NRL protocol analyzer. The first one is a reflection attack, and the second one is called the penultimate authentication flaw.

In 2003, IKEv2 is formally verified in the context of the AVISPA project [3]. The authors find that IKEv2 also suffers from the penultimate authentication flaw. However, they say that it cannot be exploited for further purposes. They propose a counter-measure anyway: the key confirmation.

In 2010, Cremers performs an extensive analysis of IKEv2 [4] using the Scyther tool. He confirms that IKEv2 suffers from the penultimate authentication flaw and, like in the AVISPA project, concludes that this vulnerability is harmless.

The penultimate authentication flaw is an execution trace of IKEv2 that violates the weak agreement authentication property. Weak agreement is a property that was first defined by Lowe in [7]. This property states that whenever an agent A has completed the protocol, apparently with an agent B, then B has previously been running the protocol, apparently with A. In the penultimate authentication flaw, A starts a session as initiator and wants to talk to C. But the intruder deviates every message A sends, to Responder B, and every message B sends, back to A (of course messages sent by B were already addressed to A, so whether the intruder does not do anything or intercept it and forward it, the attack remains). The parties proceed normally until A receives the IKE_AUTH response. The AUTH payload is not signed with the private key of C, so A does not set up a Child SA. A then sends an IKEv2 INFORMATIONAL message containing an AUTHENTICATION_FAILED notification payload. Intruder intercepts it and drops it. In the end, B has set up a Child SA with A, whereas A did not want to set up a Child SA with B. This is a violation of weak agreement for the responder.

The penultimate authentication flaw is not a full violation of the intuitive definition of authentication, because there is no actual impersonation and secrecy is still satisfied. However, we show in the next Section that the penultimate authentication flaw allows a Denial-of-Service attack.

III. The Deviation Attack

A. Preliminaries

We assume the existence of $N + M + 2$ IKEv2 parties called *(Initiator$_i$)*$_{1\leq i\leq N}$, *(Responder$_i$)*$_{1\leq i\leq M}$, *Probe*, and *Victim*. Each party may be either an IKEv2 endpoint (also called host) or a gateway. Figure 1 presents the attack scenario. On this figure and throughout the paper, we use the term *m1* as an abbreviation for IKE_SA_INIT request. We define accordingly the terms *m2*, *m3*, and *m4*.

(Initiator$_i$)$_{1\leq i\leq N}$, (Responder$_i$)$_{1\leq i\leq M}$, and Victim are connected by a network *Net*$_1$. Probe and Victim are connected by a network *Net*$_2$. Net$_1$ and Net$_2$ may be the same network and may be the Internet. We write $(I_i)_{1\leq i\leq N}$, $(R_i)_{1\leq i\leq M}$ and V the Net$_1$ IP addresses.

We call *connection* the set of SAs that were stored in a party's memory right after an IKEv2 phase 1 session (one IKE_SA_INIT exchange and one IKE_AUTH exchange). Therefore if everything went well, *connection* denotes the set of SAs containing the newly created IKE SA and its first Child SA. However, if e.g. authentication succeeded but traffic selector negotiation failed, then *connection* denotes the newly created IKE SA alone. We say that a party *handles* a connection from the moment the party installs the connection (stores it in memory) until the moment the party deletes the connection.

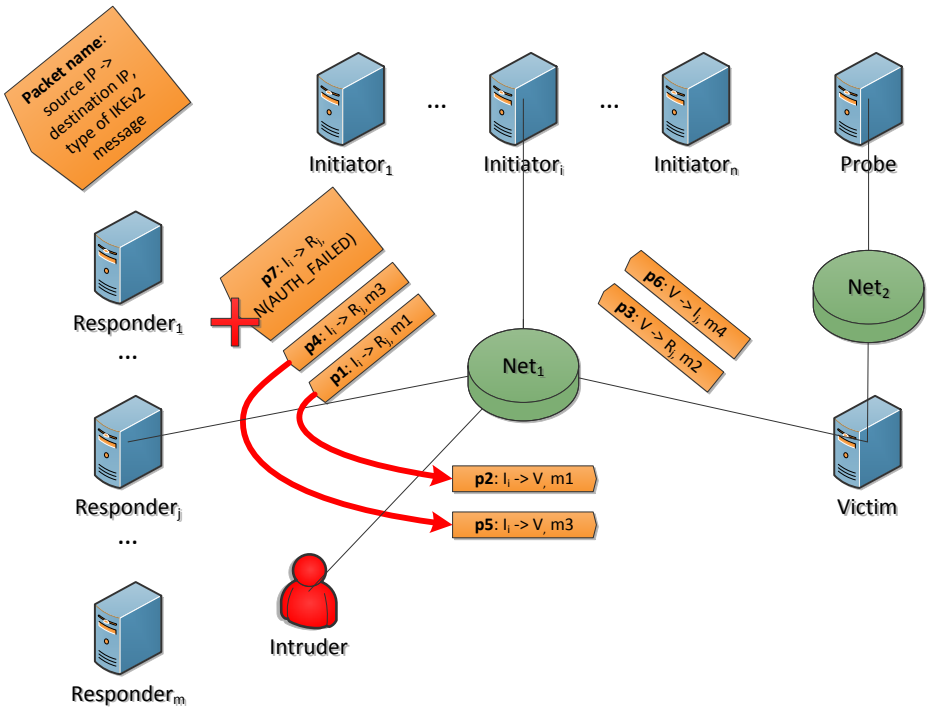

Fig. 1. Scenario of the Deviation Attack. Intruder deviates p1 and p4 and drops p7.

Let L_{app} be the application load, i.e. the amount of memory occupied by IKEv2 in Victim when there are no connection installed. We assume that L_{app} stays the same during the attack, i.e. is constant in time. Let L be the load of Victim, i.e the amount of Victim's memory that is occupied by (1) the application and by (2) connections with machines that are not Probe or Initiator machines. In practice, there could be other machines than Probe or Initiator machines initiating or removing connections during the attack, i.e. in general L is not constant in time. However, we assume for the sake of simplicity that L is constant in time. We also assume that Victim's memory is statically limited, i.e. there is no way for Victim to obtain more memory capacity during the attack. Let C be the memory capacity that was allocated to IKEv2 in Victim. We assume that C was carefully chosen and that we have $C > L$.

We say that a connection in a party's memory is *unintended* when it was not taken into account when memory was allocated to the party. Let m be the amount of memory needed to store all data related to one connection. We say that a connection in a party's memory is *stale from the beginning*, when the party has received no IKEv2 message for it since it was installed. After some time and some unanswered rekeying requests or keep-alive requests, the party removes the connection. Let S be the average time a connection stale from the beginning stays in Victim's memory.

Let *Intruder* be a machine, connected to Net_1. Let $t = 0$ be the beginning of the attack and D be the attack duration. We assume for the sake of simplicity that the Initiator parties send m1 messages to Responder parties at a constant rate σ between $t = 0$ and $t = D$. We express σ in number of IKE messages per second (and not in packets per second, in case there is fragmentation). We also assume that, at $t = 0$, no Initiator party has any SA established with Victim.

For the purpose of detecting an eventual Denial-of-Service, we make Probe regularly send m1 messages to Victim. However, we do not want Probe to affect Victim's memory occupation. We thus make Probe's connections in Victim ephemeral, i.e. Probe's connections are removed after a short time from Victim's memory. See Section IV-A for a way to achieve this for the strongSwan IKEv2 implementation.

Let $m4_t$ be the m4 message that Probe receives at time t, and let $m1_t$ be the m1 message sent by Probe that led to $m4_t$ (i.e. that belongs to the same session). Let $T_r(\sigma, t)$ be the response time of Victim to Probe at time t when throughput is σ. We define this response time as the difference between t and the time at which $m1_t$ was sent. We assume that the response time of Victim to Probe is the same as the response time of Victim to any of the $Initiator$ parties. Let T_{acc} be the maximum response time acceptable by Probe, i.e. the response time after which Probe considers that it has been denied a service. The value of T_{acc} is fixed arbitrarily.

We define the following propositions:

Req1 Intruder has the ability to intercept every IP packet sent by an Initiator party to a Responder party. When Intruder intercepts a packet, the recipient does not receive it. In

addition, Intruder can either drop the packet, or modify its destination IP address and send it to Victim.

Req2 All Initiator parties authenticate themselves using signature mode.

Req3 All Initiator parties are trusted by Victim. This means that Victim's Peer Authorization Database (see [1]) allows connections with all Initiator parties.

Req4 All m1 messages sent by Initiator parties contain at least one SA proposal (see [1]) that is acceptable to Victim.

B. Attack flow

Assume that $Req1$, $Req2$, $Req3$ and $Req4$ are satisfied. The attack proceeds as follows. Intruder intercepts all m1 messages sent by Initiator parties to Responder parties. Let us consider the implications of one specific m1 message sent by Initiator$_i$ to Responder$_j$. This message is sent in an IP packet *p1*. The message flow is shown on figure 1.

Thanks to $Req1$, Intruder intercepts the request, changes the destination IP address to V, and sends it to Victim (packet *p2*). We call this process deviation.

In response, because $Req4$ is satisfied, Victim sends an m2 message to Initiator$_i$ (packet *p3*). Initiator$_i$ receives it, and sends an m3 message to Responder$_j$ (packet *p4*). Intruder deviates the m3 message to Victim (packet *p5*).

On reception of the m3 message, authentication of Initiator$_i$ to Victim succeeds because authentication is done using signature mode ($Req2$) and because Initiator$_i$ is trusted by Victim ($Req3$). However, TS and cryptographic algorithms negotiation may fail (TS negotiation will fail in most cases). If TS and cryptographic algorithms negotiation fail, then only one IKE SA is stored. If TS and cryptographic algorithms negotiation do not fail, then one IKE SA and one Child SA are stored. According to our definition of "connection", at this point Victim has installed one connection with Initiator$_i$. Victim then sends an m4 message to Initiator$_i$ (packet *p6*).

On reception of the m4 message, Initiator$_i$ fails the authentication step, since it intended to speak with Responder$_j$, not with Victim. Initiator$_i$ thus sends an IKEv2 notification AUTHENTICATION_FAILED to Responder$_j$ (packet *p7*). Intruder intercepts the notification and drops it (thanks to $Req1$). As a result, an unintended connection was added in Victim's memory.

In this paper, we only explore memory exhaustion as a possible cause of Denial-of-Service. We state that if $Req1$, $Req2$, $Req3$ and $Req4$ are satisfied, then there is a throughput σ that allows Intruder to cause a Denial-of-Service. More formally:

Theorem III.1. *If $Req1$, $Req2$, $Req3$ and $Req4$ are satisfied, then:*

$$\{\sigma \mid \exists t \in [0, D] \mid T_r(\sigma, t) > T_{acc}\} \neq \emptyset \quad (1)$$

Proof: Since Victim's memory is statically limited, there exists a throughput σ such that, at some time during the attack, Victim is in Denial-of-Service by memory exhaustion. ■

C. Minimum throughput and DoS time

We can now define Σ_{mem} as the minimum throughput triggering a memory exhaustion.

Definition III.1.

$$\Sigma_{mem} = min(\{\sigma \mid \exists t \in [0, D] \mid T_r(\sigma, t) > T_{acc}\})$$

When $\sigma > \Sigma_{mem}$, we define $T^s_{mem}(\sigma)$ as the time at which DoS starts, and $T^e_{mem}(\sigma)$ as the time at which DoS ends.

Definition III.2. Let $\sigma > \Sigma_{mem}$. We define:

$$T^s_{mem}(\sigma) = min(\{t \in [0, D] \mid T_r(\sigma, t) > T_{acc}\})$$
$$T^e_{mem}(\sigma) = max(\{t \in [0, D] \mid T_r(\sigma, t) > T_{acc}\})$$

We confront the following theorem with the experiment in Section IV. To do so we implement the Deviation Attack and measure in different experimental setups the time at which DoS starts.

Theorem III.2. *We state that:*

$$\Sigma_{mem} = \frac{C - L}{m \times min(D, S)}$$

Furthermore, when $\sigma > \Sigma_{mem}$, memory exhaustion starts and ends at:

$$T^s_{mem}(\sigma) = \frac{C - L}{m\sigma}$$
$$T^e_{mem}(\sigma) = max(D, S)$$

Proof:

Summing up implications of all m1 messages sent by Initiator parties to Responder parties, at time t, Victim has installed $\sigma \times t$ connections in its memory. However, all these connections are stale from the beginning, so they are removed after S seconds. Therefore at time t, the number of unintended connections that are present in Victim's memory is $\sigma \times min(t, S)$. Victim thus suffers from memory exhaustion at time t if and only if:

$$\sigma \times min(t, S) > \frac{C - L}{m}$$

Since the attack last D seconds, the attack leads to a memory exhaustion if and only if:

$$\sigma \times min(D, S) > \frac{C - L}{m}$$

Which yields our expression of Σ_{mem}.

Now we assume that $\sigma > \Sigma_{mem}$. Memory exhaustion will start as soon as:

$$\sigma \times t > \frac{C - L}{m}$$

Which yields our expression of T_{mem}.

Memory exhaustion lasts until connections are removed and the attack has stopped, i.e. until $T^e_{mem}(\sigma) = max(D, S)$.

■

D. Discussion of the Deviation Attack

a) The Deviation Attack when pre-shared keys are in use: Note that we do not require that Victim authenticates itself using signature mode. In other words, even if authentication is performed asymmetrically, with Victim using signature and the Initiator machines using PSK, the attack still works.

Moreover they may be looser requirements than the ones we impose in this paper. For example the Deviation Attack is possible when the Initiator machines and Victim use a PSK to communicate with each other and when this PSK is the same as the one used by the Initiator and Responder machines to communicate with each other. If those two PSKs are different then the attack does not work.

b) Why Initiator$_i$ does not refuse message p3: In the context of a Deviation Attack, Initiator$_i$ sees that the source IP address of message p3 (see figure 1) is not the destination address of message p1. At first glance, this observation could be the witness of an odd situation. However, the IKEv2 RFC specifically says that "Incoming IKE packets are mapped to an IKE SA only using the packet's SPI, not using (for example) the source IP address of the packet" [2]. This is why Initiator$_i$ does not refuse message p3.

c) A way to obtain enough requests to deviate: For the attack to work, there need to be a sufficient rate of IKE_SA_INIT requests that are sent from the Initiator parties to the Responder parties, in a duration short enough. If this situation never arises, Intruder may have a workaround: it can drop all messages coming from the Initiator parties and going to the Responder parties, for a given time. After some unanswered IKEv2 keep-alive requests (if Dead Peer Detection is activated, see Section V-A) or some unanswered CREATE_CHILD_SA requests, the Initiator parties may consider their connections with the Responder parties as broken and may send new IKE_SA_INIT requests for each broken connection. This solution works for example if the connections are configured so as to be automatically set back up when broken (option "closeaction=restart" in strongSwan), or so as to be automatically set up when an outbound IP packet arrives (option "auto=route" in strongSwan).

d) Classification among DoS attacks: The Deviation Attack belongs to the category of slow DoS attacks (SDA). A definition of SDAs is given in [8]. An SDA is a DoS attack that requires very low amount of bandwidth. To do so, SDAs usually target a listening daemon on a host by exploiting some application layer vulnerability. Indeed, the Deviation Attack makes an IKEv2 daemon unavailable by exploiting a weakness in the application and requires very low amount of bandwidth compared to classic flooding DoS techniques.

Although it seems easier to do a simple denial of service using a high traffic load, protection from classic flooding techniques is possible by means of Intrusion Detection Systems (IDS). The Deviation Attack is much harder to detect.

IV. Attacking an IKEv2 implementation

To concretely demonstrate the Deviation Attack implications, we attack the strongSwan open-source IKEv2 implementation. Our experiment code is available at [9]. Our experiment also allows us to experimentally verify our expression T^s_{mem} that we give in theorem III.2.

A. Setup

a) Global setup: In our experiment, target IKEv2 implementation is strongSwan version 5.1.2. However throughout this Section, we will use the $software$ term when covering a topic that is not specific to strongSwan. Except when indicated, we use the default options for $software$.

To reproduce the attack, we create 3 Linux Virtual Machines (VM) representing Victim, Probe and Intruder, and N_{demo} VMs representing (Initiator$_i$)$_{1\leq i\leq N}$, where N_{demo} is a configurable parameter. We do not instantiate the Responders, since we only need their IP addresses; we do not really need the machines.

All machines are connected through the same local (virtualized) network, and they are the only ones connected to it. Using Section III-A notation, we have $Net_1 = Net_2$ and this network is local. Using a local network allows us to easily reproduce the deviation of packets by Intruder, as explained below. Moreover it ensures more control over networking propagation times and over the daemon's resource loads.

We create a Certificate Authority that we call *CA*. We generate for Probe, Victim and each Initiator a certificate signed by CA, and its associated private key. We place in Probe, Victim and each Initiator their respective certificates and private keys, along with CA's certificate.

b) Implementing the Initiators: We decided, for practical reasons, not to create the N Initiators of the generic scenario, but instead to only create N_{demo} Initiators, with each of them sending $\frac{N}{N_{demo}}$ m1 messages to the Responders. Furthermore, we make the Initiators intending to talk to only one Responder peer. This makes it easier to implement Intruder because that way Intruder only needs to spoof one IP address (see below to understand why Intruder performs spoofing). Because of these simplifications and to stay faithful to the generic scenario depicted in Section III, we had to change two options of strongSwan in Victim and the Initiators. We explain that in the experiment code's README.md file.

For strongSwan, we set the $rightid$ option in the Initiators like below. The "%" sign forces strongSwan not to send IDr in the IKE_AUTH request. We explain in Section IV-C why this is necessary.

```
rightid="%CN=responder"
```

Listing 1. The rightid option in strongSwan

The Initiators try to establish connections with Responder at a configurable rate σ. We stop the attack after some configurable time D.

c) Implementing Intruder: To reproduce the deviation of packets by Intruder, we use an ARP cache poisoning attack [10]. In this attack, Intruder sends ARP replies to all Initiators, binding its MAC address to Responder's IP. This way, all packets sent by Initiators to Responder are intercepted by Intruder. To perform this attack, we use the $arpspoof$ tool [11]

in Intruder. We then use Linux $iptables$ command to redirect the traffic towards Victim and to drop the AUTH_FAILED notification. Of course this method is only possible because we use a local network. In reality, when Net_1 is not a local network, deviation has to be made using other ways.

d) Implementing Victim: We use Linux Control groups (Cgroups) to allocate a (configurable) memory of exactly C to Victim for $software$. Note that in our setup, TS payloads sent by the Initiators are not valid propositions for Victim. Thus a connection will only consist of one childless IKE SA (containing two unidirectional SAs).

We observed in our experiment that when there is no memory left for $software$, the Linux Out-Of-Memory killer (OOM killer) of Victim kills the $software$ process. This leads to the loss of all installed SAs. This is undesirable behaviour: setting back up all SAs might take some time, meanwhile suspending the protected IP flow that used traffic SAs. To observe a memory exhaustion in our experiment, we therefore disable the Out-Of-Memory killer of the kernel and of $software$'s control group in Victim.

e) Implementing Probe: The Probe VM tries to set up a new IPsec connection every 2 seconds (configurable). For each attempt, after $T_{acc} = 5\,\text{s}$ (configurable), Probe checks if the attempt has succeeded and reports the result. Finally, as explained in Section III-A, we make Probe's connections in Victim ephemeral. To do so, we use specific options for strongSwan in Victim. We explain that in the experiment code's README.md file.

B. Verifying our DoS time expression

To experimentally verify our expression of T^s_{mem}, we measure L and m for $software$ in the context of our setup, verify that L and m are constants (i.e. that they do not depend on C or σ), and measure T^s_{mem} ($measured\ T^s_{mem}$) when tuple (C, σ) varies.

We measured σ during the experiment ($measured\ \sigma$) and observed that it was different from the σ we configured ($configured\ \sigma$). This is most probably due to virtualization. For the experimental verification of theorem III.2 not to be affected by the fact that $configured\ \sigma$ and $measured\ \sigma$ are different we thus use $measured\ \sigma$ to calculate the T^s_{mem} value predicted by theorem III.2 ($Expected\ T^s_{mem}$).

For consistency we use a warm up run. Furthermore for each tuple (C, σ) we perform the measure of T^s_{mem} (resp. σ) 10 times. We then take the average of T^s_{mem}'s (resp. σ's) measures as our value of $measured\ T^s_{mem}$ (resp. $measured\ \sigma$).

To measure m we fill Victim's memory with a high number of connections at some given throughput, and divide the memory increase by the number of connections. There are several ways to measure the amount of memory used by a process. To be consistent with how we limit memory available to $software$ (using cgroups), we take the value stored in $software$'s cgroup file $memory.usage\_in\_bytes$.

In our experiment, we have $L = L_{app}$, where L_{app} is the amount of memory occupied by IKEv2 in Victim when there is no connection installed (as defined in Section III-A). To measure L we thus simply measure the amount of memory used by $software$ when there are no connections installed.

C. Results

We observe that strongSwan differs from IKEv2's RFC on one point. When the IKE_AUTH request IDr payload does not correspond to any of the responder's identities, strongSwan notifies that no matching peer configuration has been found and cancels the IKE SA establishment. In the RFC, it is said the following: "If the IDr proposed by the initiator is not acceptable to the responder, the responder might use some other IDr to finish the exchange". In other words, if IDr does not correspond to one of its identities, the responder might install a Child SA anyway, and use some other IDr to finish the exchange.

The IKE_AUTH request IDr payload is optional, both in the RFC and in strongSwan. The behaviour adopted by strongSwan and described above thus implies the following for the Deviation Attack: When the Initiator machines run strongSwan and are configured so as to not send an IDr payload, the attack works. However, when they run strongSwan and are configured so as to send an IDr payload, the attack does not work, since IDr would be equal to a Responder machine's identity, and not Victim's.

Not sending IDr payload in an IKE_AUTH request is not an uncommon configuration, since it allows to hide the responder's identity to an active attacker. We explain this in section V-A.

Our measures of L and m confirm that L and m do not depend on C or σ (or very little). We obtain $L \approx 1\,\text{MB}$ and $m \approx 18.805\,\text{kB}$. Figure 2 shows the result of our measures of T^s_{mem}. Our measures are close to the values predicted by theorem III.2: we obtain an average relative error of 1.3% and a maximum relative error of 2.4%.

V. Counter-measures

A. Trying to protect implementations of the current protocol

Users who cannot afford to be targeted by the Deviation Attack need immediate protection. In this Section, we show that the cookie and puzzle mechanisms that were introduced in IKEv2 to resist DoS attacks are of no use against the Deviation Attack. We then note that Dead Peer Detection can be a small mitigation and consider two measures that prevent the attack but suffer from significant drawbacks: using PSK authentication and giving enough resources to Victim.

a) Existing DoS counter-measures: The cookie mechanism [2] was introduced to protect IKEv2 against a memory exhaustion due to reception of a large amount of IKE_SA_INIT requests. If this mechanism is in place, when the responder detects a large number of half-open IKE SAs, it responds to each IKE_SA_INIT request (that does not contain a cookie) with an IKEv2 INFORMATIONAL message containing a cookie. The cookie is a keyed hash of the request. The initiator then sends the same request again with the cookie added to it, and the responder verifies that the cookie and the request match. This means that the attacker needs

C (in MB)	50	50	50	50	200	200	200	200
Configured σ (in m1 messages/s)	1	5	10	30	1	5	10	30
Measured σ (in m1 messages/s)	1.0	4.8	9.5	25.6	1.0	4.9	9.5	24.1
Expected T^s_{mem} (in s)	2605	541	274	101	10582	2159	1111	438
Measured T^s_{mem} (in s)	2617	547	280	103	10771	2202	1124	445
Relative error in %	0.5	1.1	2.1	2.4	1.8	2.0	1.1	1.6

Fig. 2. Predicting and measuring DoS start time T^s_{mem} during some Deviation Attacks against strongSwan.

to keep in memory the IKE_SA_INIT requests it sends. It thus makes it more costly for an attacker to fill the half-open SA database of a gateway. However, in the Deviation Attack, cookies will be handled by the Initiator parties and not by Intruder. Activating cookies thus has absolutely no effect on Intruder's memory requirements. Therefore, it does not increase the cost of the attack in terms of memory. Note that the cookie mechanism also increases the time between the reception of an IKE_SA_INIT request by Victim and the filling of its memory with the new SA. But this neither increases the throughput of packets the attacker needs to deviate, nor the duration needed for the Deviation Attack to succeed.

The puzzle mechanism is specified in [12]. It is an improvement of the cookie mechanism. The initiator now needs to remember its request and to solve a puzzle before sending its request again. Like the cookie mechanism, the puzzle mechanism is stateless for the responder. Puzzles increase the cost of an attack in CPU power. However, in the deviation attack, it is the Initiator parties who need to solve the puzzles, not the Intruder. So puzzles, like cookies, are of no use against the Deviation Attack.

b) Dead Peer Detection: Dead Peer Detection [13] (DPD) is a mechanism left as an option in IKEv2. In fact, in strongSwan, it is not enabled by default. When DPD is in use, whenever a party sees that no traffic has recently been received on an IKE SA or any of its Child SAs, then it may send a keep-alive request to the SA's peer. If the peer does not respond, after several retransmissions, the party may remove the IKE SA and all its Child SAs from its memory. In the context of the Deviation Attack, DPD reduces S, the average time a connection stale from the beginning stays in memory. Since Σ_{mem} is inversely proportional to S, DPD makes it harder to achieve a memory exhaustion using the Deviation Attack.

However, reducing S too much can create an overload on the network, so S cannot be too low. When DPD is enabled in strongSwan, using default values for the other options, we have $S = 195s$, which multiplies Σ_{mem} by 18. Therefore, DPD only mitigates memory exhaustion.

c) Pre-Shared Key authentication: As we point out in Section III-B, the Deviation Attack is not possible when PSKs are used for authentication, provided that the Initiator machines and Victim do not share the same PSK as the Initiator machines and the Responder machines. Therefore using PSK is a counter-measure. However, PSKs and certificates do not fulfil the exact same needs. PSKs are used when the number of peers is relatively low. If this is not the case, another counter-measure must be considered.

d) Giving enough resources to Victim: One solution to the memory exhaustion is to give enough memory power to Victim to handle as many connections as there can be. Victim would then need to be given a memory of at least $L_{app} + N_t \times m$, where N_t is the number of peers that Victim trusts. This counter-measure is efficient since memory is cheap nowadays. However, IKEv2 should not rely on such a recommendation to its users.

B. Improving the protocol specification

Attempts to protect implementations of the current protocol are either not sufficient to prevent the Deviation Attack, or present significant drawbacks. For this reason, we propose two modifications of the protocol specification that deter its vulnerability to the Deviation Attack.

a) Using IDr payload: A way to modify the protocol would be to make the IDr payload in IKE_AUTH request mandatory, and to modify its processing by the responder. The appropriate behaviour would be not to install a Child SA when IDr is not *acceptable* (defined below), but instead, to cancel the establishment of the IKE SA, by sending an AUTHENTICATION_FAILED notification to the Initiator.

Let us define the term *acceptable* we used above. The IPsec and IKEv2 specifications should have a new mandatory field in the PAD. We call this field the *local ID* field. We should also rename the ID field, described in RFC 4301, Section 4.4.3, to *remote ID field*, for consistency. The local ID field would be a non-empty list of IDs. Each ID would be in the same format as the ID field (it can use wildcards, for example). When an IKE_AUTH request arrives, a PAD lookup is done. A PAD entry would match the request when both the remote ID field and the local ID field match respectively the IDi and IDr payloads.

Note that, as we explain in Section IV-C, strongSwan already implemented the local ID field. It corresponds to the $leftid$ attribute of the ipsec.conf configuration file.

The IDr modification, however, has one drawback. Assume Alice initiates an IKEv2 session with Bob, using the non-modified protocol. According to [14], IKEv2 was designed so as to hide both identities from a passive attacker and Bob's identity from an active attacker as well. It does not hide Alice's identity from an active attacker because it is Alice who reveals

its identity first using IDi payload. The attacker can learn this identity by impersonating Bob's IP address.

Now assume that we modify the protocol using the IDr method. Using the same attack where the intruder impersonates Bob, the intruder is now able to learn the identity of Bob, and even to prove that Alice intended to speak to Bob. In other words, this modification works but the responder's identity would no longer be hidden from an active attacker. This may be a problem if sensitive information can be found in the ID payload. This is often the case, as people often use Distinguished Names (DN), where the country, institution name and email address are given.

b) Adding key confirmation: There is a way to modify IKEv2-Sig that does not require to disclose the responder ID before it is cryptographically verified. We add a third exchange, called the *key confirmation* and written KEY_CONF. This exchange is exactly the same as an IKEv2 keep-alive exchange: an empty INFORMATIONAL request and an empty INFORMATIONAL response. We require that the responder installs its connection only after having received a valid KEY_CONF request and that the initiator installs its connection only after having received a valid KEY_CONF response. Key confirmation was first proposed by Basin et al. in [3] as a counter-measure to the penultimate authentication flaw. Since the Deviation Attack exploits the latter, key confirmation is a counter-measure to the Deviation Attack.

One other modification could have been to simply add a mandatory AUTH_SUCCESS message as an acknowledgement to the IKE_AUTH response. However, in IKEv2, all messages, except for error messages, exist in pairs. This is because IKEv2 is carried over UDP, so the only way to be sure that a request has been received is to wait for a response message and to set up retransmissions in case it does not arrive (until a timeout). With the KEY_CONF response, the initiator is now sure, when it installs its connection, that the responder has installed its.

Adding an exchange to the protocol can be seen as a increase of its cost, as it will take a more time to establish a connection. But a KEY_CONF message requires only a very little amount of time and computational power to generate and process, because there is no asymmetric cryptography or key derivation operations to perform. It is therefore a very efficient solution to prevent the Deviation Attack.

VI. Conclusion and future work

In this paper, we designed a novel slow Denial-of-Service attack against IKEv2: the Deviation attack. We explained its working flow, and precisely evaluated its requirements and consequences. To concretely demonstrate the attack, we successfully implemented it against the strongSwan open-source implementation. The source code to reproduce the Deviation Attack is available at [9].

The central position that IKEv2 occupies in modern infrastructures leaves no doubt that counter-measures need to be taken. We discussed the efficiency of the available means to protect implementations of the current protocol. However, none of them were complete solutions. Worse, the cookie and puzzle mechanisms that were introduced in IKEv2 to counter DoS attacks are completely ineffective against the Deviation Attack. We thus tackled the problem at a higher level and proposed two possible inexpensive modifications of the protocol, which both prevent the attack.

Finally, as we have seen, this paper outlines the importance of the weak agreement property for authentication protocols. Its violation does not necessarily imply a violation of secrecy, but we have shown that it can allow other attacks. In particular, when the protocol sets up some connection in the parties' memories, it can lead to a DoS attack. It could be interesting to verify weak agreement for TLS, SSH, and other stateful authentication protocols.

Acknowledgment

The authors would like to thank Thomas Given-Wilson for its help with writing, and Youcef Ech-Chergui for its IKEv2 expertise. In addition, the authors would like to thank the ANSSI [1] for their technical review of the paper.

References

[1] K. S. and S. K., "Security Architecture for the Internet Protocol," RFC 4301, December 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4301.txt

[2] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet key exchange protocol version 2 (IKEv2)," RFC 7296, November 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8019.txt

[3] A. Project, "Deliverable d6.2: Specification of the problems in the high-level specification language," Tech. Rep., 2003, "http://www.avispa-project.org/".

[4] C. Cremers, "Key exchange in ipsec revisited: Formal analysis of ikev1 and ikev2," in *European Symposium on Research in Computer Security*. Springer, 2011.

[5] https://www.strongswan.org/, [Online; accessed 17-February-2019].

[6] C. Meadows, "Analysis of the internet key exchange protocol using the nrl protocol analyzer," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[7] G. Lowe, "A hierarchy of authentication specifications," in *Proceedings 10th Computer Security Foundations Workshop*, Jun 1997.

[8] E. Cambiaso, G. Papaleo, and M. Aiello, "Slowdroid: Turning a smartphone into a mobile attack vector," in *2014 International Conference on Future Internet of Things and Cloud*, Aug 2014.

[9] https://gitlab.com/deviation/demo.

[10] W. contributors, "Arp spoofing," https://en.wikipedia.org/wiki/ARP_spoofing, 2018, [Online; accessed 23-January-2018].

[11] D. Song, https://linux.die.net/man/8/arpspoof.

[12] Y. Nir and V. Smyslov, "Protecting internet key exchange protocol version 2 (ikev2) implementations from distributed denial-of-service attacks," RFC 8019, November 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8019.txt

[13] G. Huang, S. Beaulieu, and D. Rochefort, "A traffic-based method of detecting dead internet key exchange (ike) peers," Internet Requests for Comments, RFC 3706, February 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3706.txt

[14] R. Perlman, "Understanding ikev2: Tutorial, and rationale for decisions," *RFC Editor*, January 2003.

[1] The ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) is the national cybersecurity agency of France